THE 65816 MICROPROCESSOR

PART 1: SOFTWARE

An 8-/16-bit successor to the 6502

WITH SEMICONDUCTOR memory prices falling, microcomputer systems are boldly going where no micros have gone before. Systems with 64K-byte main memories are common, and those with a megabyte or more are just around the corner. Since such old 8-bit standbys as the 6502, 6800, and Z80 can directly address only 64K bytes, their days as the backbone of general-purpose systems may be numbered. Indeed, the increasing number of personal computers, like Apple and Atari, using bank switching and disk simulation to expand their semiconductor memory access shows the growing severity of the address-space pinch.

Designed to relieve the constraints of limited memory, a new branch of the 6502 family directly addresses up to 16 megabytes while bringing the simplicity and execution speed of this time-tested microprocessor to 16-bit systems. The main representative of this new group is the W65SC816, which I refer to hereafter as the 65816. In emulation mode, it is pin- and software-compatible with the 6502, and toggling a single flag bit converts it to a complete 16-bit processor. In part 1 of this article I cover

(continued)

Steven P. Hendrix, an instructor pilot for the U.S. Air Force, has a B.S. in computer science and mathematics from the USAF Academy. He can be reached at Route 8, Box 81E, New Braunfels, TX 78130. His hobbies include computers and astronomy.

BY STEVEN P. HENDRIX

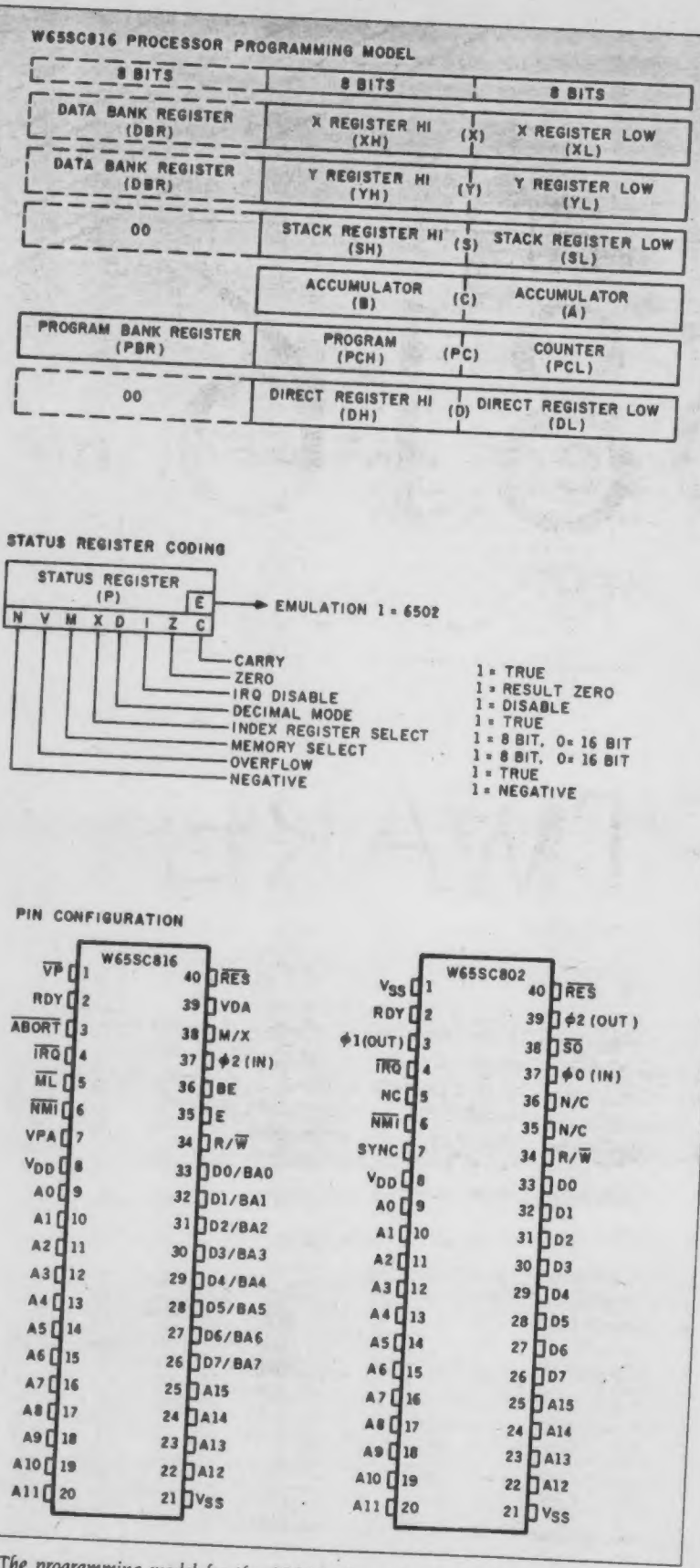

Figure 1: *The programming model for the 65816 shows that the 6502's 8-bit registers have been expanded, the program counter extended, unused bits filled, and a new register (the direct register) added.*

some of the software considerations for this processor. Next month, part 2 will address some hardware considerations. Throughout, the term "page" refers to a 256-byte memory area wherein individual location addresses differ in only the low-order 8 bits, just as for the 6502. "Bank" refers to a 64K-byte memory area with address locations that differ in the low-order 16 bits, with the upper 8 bits identical.

Overview

Figure 1 shows the programming model for the 65816. Close examination reveals the 6502 registers with extensions, plus one additional 16-bit register formerly not present (the direct register) and two 8-bit bank registers. The 8-bit registers of the 6502 have been extended to 16 bits, the 6502's 16-bit program counter has been effectively extended to 24 bits, and the previously unused bits in the status register have been filled (with an extra bit added). The accumulator and index registers still can be treated optionally as 8-bit registers by setting the appropriate status register bits. Except for the accumulator, the full 16-bit registers retain the original 8-bit names. Each half of each register uses the same original name, and they are distinguished from one another by an "H" or "L" suffix for the high- and low-order bytes, respectively. The 16-bit accumulator is newly designated as "C." The high- and low-order bytes of the accumulator are referred to as "B" and "A."

The six status register flags matching those of the 6502 retain the same meanings. The seventh 6502 flag, "B,"

(*continued on page 385*)

E = 1 (6502 Emulation)

00FFFE,F—IRQ/BRK	Hardware/Software
00FFFC,D—RESET	Hardware
00FFFA,B—NMI	Hardware
00FFF8,9—ABORT	Hardware
00FFF6,7— —	—
00FFF4,5—COP	Software

E = 0 (16-Bit Operation)

00FFFE,F—IRQ	Hardware
00FFFC,D—RESET	Hardware
00FFFA,B—NMI	Hardware
00FFF8,9—ABORT	Hardware
00FFF6,7—BRK	Software
00FFF4,5—COP	Software

Figure 2: *When the 65816 is not emulating a 6502, it has a separate vector that defines the software interrupt's location and relieves the routine from having to determine which type of interrupt invoked it.*

(continued from page 126)
is now "X." On the 6502 (and on the 65816 in its 6502 emulation), X indicates a BRK (Break) instruction. A software interrupt, it transfers control to the same routine as does the hardware interrupt, IRQ (masked by flag "I"). When not emulating a 6502, the 65816 adds a separate vector (see figure 2) that defines the software interrupt routine's location. This vector relieves the routine from having to determine which type of interrupt invoked it.

The new "M" flag, as well as X, is used when the 65816 is not emulating a 6502. They both determine whether operations treat the accumulator and index registers as 8- or 16-bit entities. The M flag, when set to "1," indicates that the accumulator is to be treated as an 8-bit register; when set to 0, the accumulator is 16 bits long. The X flag has the same effect on the index (X and Y) registers.

The added "E" flag determines whether the 65816 emulates a 6502 or acts as a 16-bit processor. Since the status register is defined as having only 8 bits, the E flag is conceptually an extension of the C flag. While E cannot be directly tested, set, or cleared, XCE (exchange C and E flags) permits swapping with the C flag, which can.

The additional 16-bit register, called the "direct register," determines placement of the "direct page" in bank zero. As on the 6502, one special page in memory is designated for the most frequently used or most time-critical portions of a program. Because the address within this page can be completely specified in a single byte, instructions can be shorter. This arrangement results in smaller programs, and, since the processor fetches one less byte per instruction, execution is faster. On the 6502, the direct page (also called "zero page") resides permanently at the beginning of memory; that is, the high-order byte of the address is 0. On the 65816, the direct page can appear anywhere in bank zero, even across normal page boundaries. This feature permits more flexibility in context switching and sharing of data between subroutines or coroutines. Also notable, if the low-order byte of the direct register (DL) is nonzero, the 65816 needs an extra cycle for every instruction that has any form of direct page addressing. The additional cycle is used to add DL to an offset. If DL is zero, the 65816 has special circuitry to skip that cycle. Except in special cases, therefore, the direct page should begin on a page boundary. The direct register can be accessed by transferring data to or from either the C accumulator or the stack.

The two new 8-bit bank registers give the 65816 its ability to access a full 16 megabytes of memory. As shown in figure 1, the program bank register (PBR) is an extension of the program counter. A program can read the register's contents only after pushing them onto the stack and then loading them into another processor register. PBR is modified by "long" forms of the jump and jump-to-subroutine instructions. Although its contents can be pushed onto the stack, PBR cannot be loaded except when it is also loading the program counter. If PBR were ever loaded independently, the next instruction after the one that loaded it would be fetched from the "following" location—but in a different bank. This unfamiliar instruction, in turn, would most likely lead the processor off into never-never land.

In some addressing modes, such as direct page addressing, the data bank registers explicitly specify the bank number. In any case, where the bank number is not laid down precisely, the data bank registers assign one. Strictly speaking, they are not extensions of the index registers, except in the sense that they help form those addresses that use the indexed addressing modes. The data bank registers are accessed through the stack and can be modified by the block move instructions described next.

New Instructions

In "The CMOS 6502" (December 1983 BYTE, page 443), I pointed out that the newer version added a number of instructions to those of the NMOS (negative-channel metal-oxide semiconductor) 6502. Those instructions filled gaps in the instruction set and rounded it out for the 6502.

The 65816 goes a step further by adding some nice-to-have instructions plus some others needed for the change to 16 bits and the addition of new registers. The op (operation) codes corresponding to hexadecimal codes 7*x* and F*x* ("set and reset individual bits in page zero" and "branch if bit set or reset in page zero") are present only on the Rockwell version of the 65C02. Western Design Center, the designer of both the 65C02 and the 65816, does not specify them. On the 65816, Western reserves the 7*x* and F*x* equivalents for "long" addressing modes of existing instructions. A description of the long addressing modes follows.

Figure 3 shows the complete instruction set of the 65816. The hexadecimal code for each op code is given by the digit at the left end of the row (most significant digit) and the digit at the top of the column (least significant digit). The op-code mnemonic is three capital letters, followed by the addressing mode in lowercase. The number at the lower left of each box gives the number of bytes for the op code plus its operand(s) (address or data). In the case of the immediate addressing mode with 16-bit data, one additional byte is required. The number in the lower right of each box gives the basic number of clock cycles required for the instruction.

Additional cycles are required for a variety of cases, since this figure is merely the number of cycles for executing that instruction in the fastest case. For example, add one cycle for instructions using the direct register for part of an address if DL is not zero; add a cycle for read or write operations using 16-bit data; add two cycles for read-modify-write instructions, such as shifts, using 16-bit data; add one cycle for branch instructions (including BRA) when the branch is taken, and another if the branch crosses page boundaries in the 6502 emulation mode; add one cycle for indexed addressing modes where the indexing causes a page boundary crossing; and add one cycle for REP or SEP (reset or set status register flags) instructions using the immediate addressing mode.

The NMOS 6502 allows several types of conditional program branches but does not include an unconditional form. Since this is the only type of position-independent program transfer on the 6502, some programs use a sequence such as SEC, BCS (set the carry flag, branch if carry set, respectively) to achieve an unconditional branch. The 65816 instruction BRA (branch relative always) fills this gap in the instruction

(continued)

Table 4. W65SC816 Microprocessor Op Code Matrix

MSD \ LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	MSD
0	BRK s 2 8	ORA(d,x) 2 6	COP s 2*8	ORA sr 2*4	TSB d 2•5	ORA d 2 3	ASL d 2 5	ORA(dl) 2*6	PHP s 1 3	ORA imm 2 2	ASL acc 1 2	PHD s 1*4	TSB a 3•6	ORA a 3 4	ASL a 3 6	ORA al 4*5	0
1	BPL r 2 2	ORA(d),y 2 5	ORA (d) 2•5	ORA(sr),y 2*7	TRB d 2•5	ORA d,x 2 4	ASL d,x 2 6	ORA(dl),y 2*6	CLC imp 1 2	ORA a,y 3 4	INC acc 1•2	TCS imp 1*2	TRB a 3•6	ORA a,x 3 4	ASL a,x 3 7	ORA al,x 4*5	1
2	JSR a 3 6	AND(d,x) 2 6	JSL al 4*8	AND sr 2*4	BIT d 2 3	AND d 2 3	ROL d 2 5	AND(dl) 2*6	PLP s 1 4	AND imm 2 2	ROL acc 1 2	PLD s 1*5	BIT a 3 4	AND a 3 4	ROL a 3 6	AND al 4*5	2
3	BMI r 2 2	AND(d),y 2 5	AND(d) 2•5	AND(sr),y 2*7	BIT d,x 2•4	AND d,x 2 4	ROL d,x 2 6	AND(dl),y 2*6	SEC imp 1 2	AND a,y 3 4	DEC acc 1•2	TSC imp 1*2	BIT a,x 3•4	AND a,x 3 4	ROL a,x 3 7	AND al,x 4*5	3
4	RTI s 1 7	EOR(d,x) 2 6	WDM RESERVED*	EOR sr 2*4	MVP xyc 3*7	EOR d 2 3	LSR d 2 5	EOR(dl) 2*6	PHA s 1 3	EOR imm 2 2	LSR acc 1 2	PHK s 1*3	JMP a 3 3	EOR a 3 4	LSR a 3 6	EOR al 4*5	4
5	BVC r 2 2	EOR(d),y 2 5	EOR (d) 2•5	EOR(sr),y 2*7	MVN xyc 3*7	EOR d,x 2 4	LSR d,x 2 6	EOR(dl),y 2*6	CLI imp 1 2	EOR a,y 3 4	PHY s 1•3	TCD imp 1*2	JMP al 3*4	EOR a,x 3 4	LSR a,x 3 7	EOR al,x 4*5	5
6	RTS s 1 6	ADC(d,x) 2 6	PER s 3*6	ADC sr 2*4	STZ d 2•3	ADC d 2 3	ROR d 2 5	ADC(dl) 2*6	PLA s 1 4	ADC imm 2 2	ROR acc 1 2	RTL s 1*6	JMP (a) 3 5	ADC a 3 4	ROR a 3 6	ADC al 4*5	6
7	BVS r 2 2	ADC(d),y 2 5	ADC(d) 2•5	ADC(sr),y 2*7	STZ d,x 2•4	ADC d,x 2 4	ROR d,x 2 6	ADC(dl),y 2*6	SEI imp 1 2	ADC a,y 3 4	PLY s 1•4	TDC imp 1*2	JMP(a,x) 3•6	ADC a,x 3 4	ROR a,x 3 7	ADC al,x 4*5	7
8	BRA r 2•2	STA(d,x) 2 6	BRL rl 3*3	STA sr 2*4	STY d 2 3	STA d 2 3	STX d 2 3	STA(dl) 2*6	DEY imp 1 2	BIT imm 2•2	TXA imm 1 2	PHB s 1*3	STY a 3 4	STA a 3 4	STX a 3 6	STA al 4*5	8
9	BCC r 2 2	STA(d),y 2 6	STA (d) 2•5	STA(sr),y 2*7	STY d,x 2 4	STA d,x 2 4	STX d,y 2 4	STA(dl),y 2*6	TYA imp 1 2	STA a,y 3 5	TXS imp 1 2	TXY imp 1*2	STZ a 3•4	STA a,x 3 5	STZ a,x 3•5	STA al,x 4*5	9
A	LDY imm 2 2	LDA(d,x) 2 6	LDX imm 2 2	LDA sr 2*4	LDY d 2 3	LDA d 2 3	LDX d 2 3	LDA(dl) 2*6	TAY imp 1 2	LDA imm 2 2	TAX imp 1 2	PLB s 1*4	LDY a 3 4	LDA a 3 4	LDX a 3 4	LDA al 4*5	A
B	BCS r 2 2	LDA(d),y 2 5	LDA(d) 2•5	LDA(sr),y 2*7	LDY d,x 2 4	LDA d,x 2 4	LDX d,y 2 4	LDA(dl)y 2*6	CLV imp 1 2	LDA a,y 3 4	TSX imp 1 2	TYX imp 1*2	LDY a,x 3 4	LDA a,x 3 4	LDX a,y 3 4	LDA al,x 4*5	B
C	CPY imm 2 2	CMP(d,x) 2 6	REP imm 2*3	CMP sr 2*4	CPY d 2 3	CMP d 2 3	DEC d 2 5	CMP(dl) 2*6	INY imp 1 2	CMP imm 2 2	DEX imm 1 2	WAI imp 1*3	CPY a 3 4	CMP a 3 4	DEC a 3 6	CMP al 4*5	C
D	BNE r 2 2	CMP(d),y 2 5	CMP (d) 2•5	CMP(sr),y 2*7	PEI s 2*6	CMP d,x 2 4	DEC d,x 2 6	CMP(dl),y 2*6	CLD imp 1 2	CMP a,y 3 4	PHX s 1•3	STP imp 1*3	JML (a) 3•6	CMP a,x 3 4	DEC a,x 3 7	CMP al,x 4*5	D
E	CPX imm 2 2	SBC(d,x) 2 6	SEP imm 2*3	SBC sr 2*4	LPX d 2 3	SBC d 2 3	INC d 2 5	SBC(dl) 2 6	INX imp 1 2	SBC imm 2 2	NOP imp 1 2	XBA imp 1*3	CPX a 3 4	SBC a 3 4	INC a 3 6	SBC al 4*5	E
F	BEQ r 2 2	SBC(d),y 2 5	SBC (d) 2•5	SBC(sr),y 2*7	PEA s 3*5	SBC d,x 2 4	INC d,x 2 6	SBC(dl),y 2*6	SED imp 1 2	SBC a,y 3 4	PLX s 1•4	XCE imp 1*2	JSR(a,x) 3*6	SBC a,x 3 4	INC a,x 3 7	SBC al,x 4*5	F

* New W65SC816 Op Codes
• W65SC02 Op Codes

Figure 3: *The 65816's instruction set with hexadecimal equivalents, mnemonics, addressing-mode number of bytes for the op code and operand, and the basic number of clock cycles required for the instruction.*

set. Not only is BRA position independent, it requires 1 less byte than the alternative JMP (unconditional jump) instruction. It is limited to destinations within 126 bytes before and 129 bytes after the branch instruction, but this range is frequently wide enough for loops within a program.

Because the 6502 has a limited number of registers, the index registers are frequently turned to general-purpose uses. Since there are no instructions to directly transfer data between index registers and the stack or between each other, the accumulator sometimes is used to mediate transfers. The sequence PHA (push the A register on the stack), TXA (transfer the contents of the X register to A), PHA, TYA (transfer the contents of Y to A), PHA is common in subroutines that must preserve all processor registers for the calling routine. The sequence for restoring the registers is equally clumsy. Also, the data in the A register is destroyed, and it is tricky for the subroutine to recover this data from the stack without destroying data for the index registers. The 65816 adds the instructions PHX, PHY (push index registers to stack), PLX, PLY (pull index registers from stack), TXY, and TYX (transfer X to Y and Y to X, respectively). This sequence lets a subroutine save just those registers it needs to use and restore, or save them in a different sequence on the stack so that the data is available later in the correct sequence.

The 6502 has found applications both in general-purpose systems and in special applications, such as printer controllers. Most general-purpose applications don't often need the ability to set or reset specific bits of memory or input/output ports. Such dedicated applications as printer or appliance controllers, on the other hand, frequently rely heavily on this ability.

Multiple-processor systems need the

ability to set and clear semaphores to arbitrate the use of resources such as memory and peripherals. Furthermore, they must be able to do it in such a way as to change a bit and then have access to its previous state. The requirement is made even more demanding by the stipulation that no other processor can also get to the semaphore during such access. The TRB (test and reset bits) and TSB (test and set bits) instructions provide this capability. Bit 7 of the addressed memory location is copied to the C flag, and bit 6 is transferred to the V flag. TRB then computes the logical AND of the memory data and the inverse of the accumulator data, while TSB computes the logical OR of the memory data and the accumulator. Either instruction then stores the result back in the memory location and sets the Z flag to indicate whether or not the result was 0. The effect is to set or clear the bits corresponding to "1" bits in the accumulator, while loading the N and V flags with the previous state of bits 6 and 7 of memory. Two similar instructions, REP and SEP, use a byte of immediate data to determine which status register bits to change. Using immediate data is more economical than adding distinct instructions for controlling each such bit. There is still no instruction for complementing any of the flags.

Eight new instructions permit new types of stack accesses. Five of the eight transfer registers to and from the stack: PHB and PLB for the data bank register, PHD and PLD for the direct register, and PHK for the program bank register. Again, there is no instruction to pull just the program bank register from the stack.

The three other new stack instructions permit pushing computed data onto the stack and then later pulling it back into one of the processor registers or accessing it through new stack-relative addressing modes. The simplest of the three is PEA (push effective absolute address). This instruction pushes the third byte and second byte of the instruction, in that order, onto the stack. The name implies that the instruction is meant to be used to pass a data block's address to a subroutine. However, it might also be used to pass any fixed 16-bit data. The effect is the same as loading the accumulator with 16-bit data and then pushing the accumulator to the stack, except that data in the accumulator is left intact and PEA executes faster.

The PEI (push effective indirect address) instruction is only slightly more complex than the PEA instruction. The second byte of the instruction is used as an offset into the direct page (pointed to by the direct register), and the two bytes of data at that and the next-higher memory location are pushed onto the stack. The effect is to save the contents of two memory locations in the direct page, just as if they

(continued)

made up a 16-bit processor register.

PER (push effective program-counter-relative address) passes the address of a data block to a subroutine. The data is then stored within the calling routine, and the address is given as an offset from the PER instruction's location. The calling routine treats the second and third bytes of the instruction as a 16-bit quantity and adds them to the program counter. The 16-bit result is the data to be pushed on the stack. PEA and PER both pass data stored in the calling routine's code. While PEA can be used to pass a single 16-bit quantity to a subroutine, PER passes the address of a whole block of data.

Since the 65816 has several more internal registers than the 6502, it needs some new instructions for transferring data inside itself. Transfers between the X and Y index registers were covered above. The TCD (transfer accumulator C to the direct register) and TDC (transfer D to C) instructions provide the only way to set or read the direct register without using the stack. The 65816's TCS (transfer accumulator C to stack pointer) and TSC (transfer S to C) instructions ease the 6502's bottleneck of being able to access the stack pointer only through the X register. XBA (exchange B and A) lets you flip-flop the upper and lower 8-bit halves of the accumulator to help it handle mixed 8- and 16-bit data. The Z and N flags are set during the exchange to indicate the new status of the accumulator. The XCE (exchange carry flag with emulation flag) is needed because the emulation (E) flag does not actually belong to the status register proper.

The RTL (return from subroutine—long) instruction is needed because a new absolute long addressing mode has been applied to the JSR (jump to subroutine) instruction. In addition to restoring the program counter, RTL also restores the program bank register from the stack. In keeping with the convention that higher-order bytes appear higher in memory, RTL pushes the program bank register before and pulls it after the normal program counter. RTI (return from interrupt) is always treated by the program as a long return, since interrupts, by their very nature, must specify the full address of the interrupt routine, possibly changing the program bank register in the process.

Several new instructions deal with control of the system bus and clocks. The COP (coprocessor) instruction supports coprocessors and acts like a software interrupt, except with its own vector. The STP (stop the clock) instruction stops the system clock with the phase two clock high. This state can be released only by a reset, which then functions as a high-priority interrupt, with the stack set up correctly for the RTI instruction to resume processing with the instruction following the STP instruction. The WAI (wait for interrupt) instruction stops execution and waits for either the maskable or nonmaskable interrupt (IRQ or NMI). One additional code (WDM) is defined as a no-operation instruction, but it is reserved for future use. The mnemonic comes from the design engineer's initials.

The new block-move instructions, MVP and MVN, are reminiscent of the block moves of the Z80 processor, except that they can be used to move a block of memory anywhere within the full 16-megabyte address space. MVP (move preceding) is meant to move data higher in memory. The move starts at the top end of the block and proceeds downward. This procedure avoids overwriting data that has yet to be moved in cases where the old and new block locations overlap.

MVN (move next) is similarly intended for moving data. In this case, however, data is moved to a position that is lower in memory, starting at the low end of the block and working upward through it. The second byte of the instruction gives the bank address of the destination memory area and the third byte gives the bank address of the old location in memory. The X register contains the low-order 16 bits of the first byte's current address. The Y register contains the low-order 16 bits of the new address. The C accumulator contains the number of to-be-moved bytes minus one. Each byte moved requires seven clock cycles, and the move can be interrupted and resumed through the normal interrupt mechanisms. After completion, the data bank register contains the destination memory area's bank address. This is an important consideration since the program will normally need to operate on the data after moving it. The X and Y registers are incremented or decremented as the move proceeds. As a result, the Y register will hold the 16 low-order address bits of the next byte beyond (above or below) the new area. The A register contains hexadecimal FFFF. The 65816 boasts versions suitable for clock rates from 1 to 10 MHz. This capacity allows a system using the slowest clock to move a complete 64K-byte data block (characters to a very high resolution memory-mapped video display, for example) in just under half a second. With a 10-MHz clock, the same transfer would take somewhat less time than a complete screen refresh (standard monitor with interlaced scanning).

Addressing Modes

The 65816 includes all the addressing modes of the standard 6502 plus a number of new ones. The existing addressing modes were extended to allow for 24-bit addresses and 16-bit data where appropriate. Internal operations employing implied addresses take the data width corresponding to the specific registers being used. Most of the 6502 addressing modes now have separate long addressing modes, allowing them to use either a 16-bit address within the bank specified by the data bank pointer or a 24-bit address within the full 16-megabyte address space.

The immediate mode of the 6502 assumes that one of the operands is an 8-bit quantity stored immediately after the op code. For operations on 16-bit registers, the 65816 lets you store 16-bit data in the two bytes following the operation.

The addressing modes—absolute, indirect indexed, and absolute indexed with X (but not Y)—have new long forms as well as standard forms. Their standard addressing modes specify the low-order 16 bits of the address; the high-order 8 bits come from the data bank register. The zero page designation on certain modes has been replaced by direct page. The change is consistent with this special page's new mobility as granted by the direct register but leads to such double-talk as "direct indirect indexed" for an addressing mode description. To translate, keep in mind that "direct" always refers to the direct register or page, "indirect" refers to the memory location(s) holding a further address rather than data, and "indexed" means adding an index register to the

address formed up to that point. Also, the name of the addressing mode describes, from left to right, the order in which the operations are performed. Thus, the example "direct indirect indexed" means that the instruction's operand specifies a direct-page location (direct), which contains only an address (indirect), and to which the Y register is added (indexed). Adding the Y register forms the data's final address for that particular instruction. Other addressing modes use these terms in different sequences and in combination with other, more self-explanatory terms. For all cases, the same general idea holds.

The branch, jump, and jump-to-subroutine instructions each retain their old 6502 addressing modes. The jump and jump-to-subroutine instructions also have new long addressing modes that load both the program counter and the program bank register with new values. In addition, both jump and jump-to-subroutine have new indexed addressing modes that use the X register as the index of a subroutine address table. The unconditional branch instruction's long addressing mode specifies a 16-bit displacement from the current location. This displacement can be positive or negative and wraps around within the current program bank. Conditional branches have no analogous long addressing mode.

Expanded stack addressing modes complement the new stack manipulation instructions. With parameters passed to it on the stack, a subroutine can read or modify them as easily as it can access those parameters stored in a fixed block of memory. It can pop or push parameters as before and can also access them without moving the stack pointer. All it has to do is specify a displacement ranging from 0 to 255 bytes above the stack pointer. Using the "stack relative indexed" addressing mode, a subroutine can use a stack parameter as the table address to be indexed by the Y index register. Since the stack pointer can now be transferred directly to or from the accumulator, a simple arithmetic operation can allocate or delete large stack areas. Combined with the long relative branch, the new stack-addressing modes herald the 6502 family's move toward position-independent, reentrant, and even recursive routines.

With its enhanced instruction set, new addressing modes, expanded interrupt handling, and 24-megabyte address space, the 65816 enters the 16-bit arena on equal footing with the 8086 and 68000. Its one handicap is its lateness, but that might be offset by the familiar, compact, and powerful 6502 instruction set that it brings along. Next month I will show you some hardware enhancements that ease the system designer's burden when using this 6502 cousin in boxes ranging from dedicated controllers to multiuser, multiprocessor systems. ■